We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

20

Reliability Prediction and Sensitivity Analysis of Web Services Composition

Duhang Zhong, Zhichang Qi and Xishan Xu
School of Computer Science, National University of Defense Technology
P.R.China

1. Introduction

Web services are emerging as a major technology for deploying automated interactions between distributed and heterogeneous applications. It aims at the transparent integration of Web applications, based on XML-related standards (F.Curbera et al., 2002). Until now, many research efforts have been made in the field of Web services composition. Moreover, many composition languages have recently emerged, including BPEL, BPML or ebXML, these languages focus on tracking and executing collaborative business processes by business applications.

An important issue for business process built in this way is how to assess the degree of trustworthiness, especially their performance and dependability characteristics. In this paper we focus on reliability aspects, and propose an approach to predict the reliability of web services composition.

Stochastic Petri Nets (SPNs) can be used to specify the problem in a concise fashion and the underlying Markov chain can then be generated automatically. In this paper, we propose the usage of CSPN model, an extension of stochastic Petri nets as a solution to the problems of predicting the reliability of web service composition. The choice of Petri nets was motivated by the following reasons: (a) Petri nets are a graphic notation with formal semantics, (b) the state of a Petri net can be modelled explicitly, (c) the availability of many analysis techniques for Petri nets.

The remainder of this paper is organized as follows. Section 2 provides general information about BPEL and stochastic Petri net. In Section 3 we describe our reliability prediction model and propose an approach to transform BPEL process into CSPN model. Section 4 discusses the result of this mapping on an example BPEL process models. Next, Section 5 discusses the sensitivity analysis of the reliability prediction model. Finally, we discuss the related works and conclude this paper.

2. Background

2.1 BPEL

BPEL, also known as BPEL4WS, build on IBM's WSFL (Web Services Flow Language) and Microsoft's XLANG (Web Services for Business Process Design). It combines the features of a block structured process language (XLANG) with those of a graph-based process language (WSFL). BPEL is intended to describe a business process in two different ways: executable

Source: Petri Net,Theory and Applications, Book edited by: Vedran Kordic, ISBN 978-3-902613-12-7, pp. 534, February 2008, I-Tech Education and Publishing, Vienna, Austria

and abstract processes. An abstract process is a business protocol specifying the message exchange behaviour between different parties without revealing the internal behaviour of any of them. An executable process specifies the execution order between a number of constituent activities, the partners involved, the message exchanged between these partners, and the fault and exception handling mechanisms (Axel Martens, 2005).

A composite service in BPEL is described in terms of a process. Each element in the process is called an activity. BPEL provides two kinds of activities: primitive activities and structured activities. Primitive activities perform simple operations such as receive (waiting for a message from an external partner), reply (reply a message to a partner), invoke (invoke a partner), assign (copying a value from one place to another), throw (generating a fault), terminate (stopping the entire process instance), wait (wait for a certain time), empty (do nothing).

To enable the representation of complex structures, a structured activity is used to define the order on the primitive activities. It can be nested with other structured activities. The set of structured activities includes: sequence (collection of activities to be performed sequentially), flow (specifying one or more activities to be performed concurrently), while (while loop), switch (selects one control path from a set of choices), pick (blocking and waiting for a suitable message). The most important structured activity is a scope. A scope is a means of explicitly packaging activities together such that they can share common fault handling and compensation routines. It consists of a set of optional fault handlers (exceptions can be handled during the execution of its enclosing scope), a single optional compensation handler (inverse some effects which happened during the execution of activities), and the primary activity of the scope which defines its behaviour. (Sebastian Hinz et al., 2005)

The sequence, flow, switch, pick and while constructs provide a means of expressing structured flow dependencies. In addition to these constructs, BPEL provides another construct known as control links which, together with the associated notions of join condition and transition condition, support the definition of precedence, synchronization and conditional dependencies on top of those captured by the structured activity constructs. A control link between activities A and B indicates that B cannot start before A has either completed or has been skipped. Moreover, B can only be executed if its associated join condition evaluates to true, otherwise B is skipped. An activity X propagates a positive value along an outgoing link L if and only if X was executed (as opposed to being skipped) and the transition condition associated to L evaluates to true. Transition conditions are Boolean expressions over the process variables. The process by which positive and negative values are propagated along control links, causing activities to be executed or skipped, is called dead path elimination.

2.2 Stochastic Petri nets

Petri Nets (PNs) is a modeling formalism used for the analysis of a wide range of systems coming from different domains (e.g., distributed computing, telecommunication, control systems, workflow management) and characterized by situations of concurrency, synchronization, causality and conflict (Simona Bernardi, 2003). A PN is basically characterized by places, transitions and weighted arcs defining its structure and it is graphically represented by a directed bipartite graph in which places are drawn as circles, transitions are drawn as bars, input and output arcs are drawn as arrows and inhibitor arcs are drawn as circle headed arrows.

In the original definition of PNs do not include time concepts; temporal specification in PN models was introduced with different approaches, mostly by associating a delay to transitions. In particular, in Stochastic Petri Nets (SPNs) transitions firing delays are exponentially distributed random variables.

Generalized Stochastic Petri Nets (GSPNs) (Marsan A et al., 1995) are an extension of SPNs proposed by M. Molloy in which stochastic timing is mixed with deterministic null delays. In a GSPN model, there are two types of transitions: immediate transitions and timed transitions. Immediate transitions are fired in zero time and used to model logical actions or activities that require a negligible time; while timed transitions are characterized by exponentially distributed firing delays.

Definition 2.1 A GSPN model is a 6-tuple $(P, T, F, W, M_0, \lambda)$, where P is a finite set of places. T is a finite set of transitions partitioned into two subsets: T_I (immediate) and T_D (timed) transitions, where transitions $t \in T_D$ are associated with rate λ. $F \subseteq (P*T) \cup (T*P)$ is a set of arcs. $M_0=\{m_{01}, m_{02}, ..., m_{0k}\}$ is an initial marking. $W : T \to R$ is a function defined on the set of transitions. Timed transitions are associated with priority zero, whereas all other priority levels are reserved for immediate transitions. The immediate transitions are drawn as thin bars, while the timed transitions are drawn as rectangles.

SRNs are an extension of GSPNs (Gianfranco Ciardo et al., 1992), i.e., they include all the features of GSPNs and many more such as guards, timed transition priorities, variable cardinality arcs, halting conditions, and reward rates etc. None of these extensions enhance the modelling power since every SRN model can be converted to a continuous-time Markov chain(CTMC) and CTMCs are isomorphic to GSPNs(although SRNs allow calculation of some reward-based measures which are not possible through GSPNs). Thus any system that can be modelled by a SRN can also be modelled by a GSPN. However, SRNs and GSPNs differ in the conciseness of model specification. SRNs permit a much more concise description of system dependability than GSPNs do.

3. Reliability prediction using CSPN models

3.1 The CSPN model

A basic principle of the SOC paradigms is that each service composition can iteself become a service that can be recursively used in other services' composition. So we distinguish two kinds of service (Vincenzo, 2005):

- *Atomic services* don't require the services of any other resources to perform their tasks. They include, for example, the services offered by basic processing and communication resources but also the services offered by self-contained software components strictly tied to a particular computing platform.
- *A composite service* is realized as a composition of other dynamically selected services that it requires to perform its tasks.

From the reliability prediction viewpoint , the basic difference between these two service types is that the atomic service provider can publish complete reliability information that's directly useful in a service composition's reliability analysis, whereas a composite service provider is only aware of reliability information concerning the part of service implemention under its direct control. The provider must combine this information with the reliability of the other dynamically selected services to get overall service reliability. Hence, to support a service composition's reliability prediction, composite service must provide their service-usage profile, a description of the generated pattern of external service requests

As pointed out by Jens Happe (Jens Happe&Viktoria Firus, 2005), most of the reliability prediction models are based on Markov models. A Markov model can be seen as a finite state machine, whose transitions are annotated with a probability of taking the transition from its source states. These models can be appropriate when dealing with sequential systems. However, as soon as a concurrent or parallel software system (e.g. web service composition) has to be analyzed, different influences come into play, which can hardly be expresses by finite state machines or the corresponding Markov model.

To represent the service-usage profile of web services composition, we propose the Composite Service Process Net model(CSPN) based on the Stochastic Petri Net. In the CSPN model, the basic activity is represented by timed transition, the structure is represented by the immediate transitions and firing rules.

Definition 3.1 A CSPN model is a 4-tuple $\Sigma=(N,\Omega,s,t)$,where:

- N is a GSPN or SRN;
- Ω is the set of external services' operation;
- s represents the starting place of process, a token in the place indicates the service is ready to start.
- t represents the finished place of process, a token in the place indicates the service is terminated.

In the CSPN model, we can distinguish two types of transitions: operation transition represents the invoke of external services; while internal transitions represent the internal activity.

3.2 Transformation of BPEL process into CSPN model

The transformation details of primitive and structured activities into CSPN can be illustrated by these examples in Fig.1. Each primitive activity is represented by one transition. A sequence of activities is represented by the sequential concatenation of one Petri net pattern for each of the activities. A flow activity provides parallel execution and synchronization of activities, two immediate transitions are used to split the control flow into concurrent threads and join them at the end. A switch activity supports conditional routing between activities; the probability of each branch is represented by the weight of immediate transition. BPEL's while activity supports iterative performance of a specified iterative activity. The iterative activity is performed until the given Boolean while conditions no longer holds true. A pick activity exhibits the conditional behaviour where decision making is triggered by external events or system timeout. It has a set of branches in the form of an event associated with it, and exactly one of the branches is selected upon the occurrence of the event associated with it.

Control links are non-structural constructs used to express control dependencies between activities. Each activity within a flow can be source and/or target of several links. Fig 2 depicts the mapping of a linked activity X. The activity X has two incoming and two outcoming links. Each link is transformed into two places lst("link status true") and lsf("link status false") reflecting the Boolean value of that link. Before the activity X can be executed, all incoming links have to be evaluated with respect to the join condition. In Fig 2, the subnet enclosed in the box labelled specifies the mapping of incoming links to activity X, Here the join condition "AND" is defined. In general, each join condition over n links could be expressed by immediate transitions. The subnet enclosed in the box labelled specifies the mapping of outgoing links from activity X, once it is complete; it is ready to evaluate transition conditions to determine the link status for each of the outgoing links.

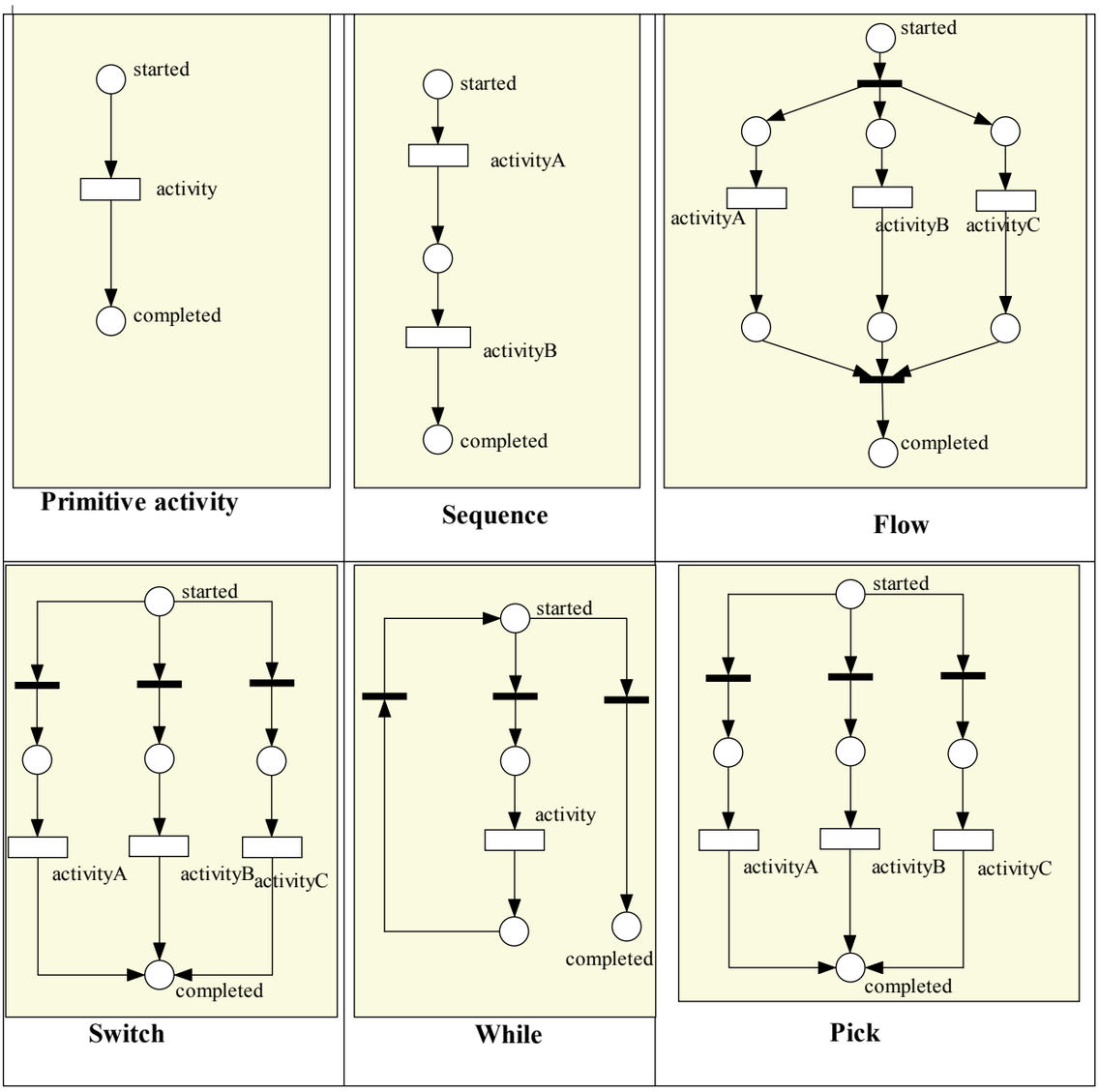


Fig 1. Transformation of BPEL into CSPN

If the join condition evaluates to true, the activity X can start as normal. Otherwise, a fault called join failure occurs. A join failure can be handled in two different ways, as determined by the suppressJoinFailure attribute associated with activity X. If this attribute is set to "yes", the join failure will be suppressed, as modelled by transition "sjf"("suppress join failure"). In this case, the activity will not be activated and the status of all outgoing links will be set to FALSE. If the activity lies on a path of an alternative branch that was not chosen, all outgoing links have to be set to false too. In that case, the activity will not be activated. Instead, it will get a token on the place negLink("propagate negative link values"). The negLink pattern sets all outgoing links to FALSE and propagates the negLink token towards the embedded activities. This is known as dead path elimination.

3.3 Computing the reliability prediction

In the next step, we annotate the CSPN model with the dependability attributes, and derive the reliability prediction of web service composition. There are three kinds of dependability attributes to be annotated:

- For every timed transition which represents the execution of a primitive activity, we annotate the execution time of the activity, which is assumed to be exponentially distributed with mean.
- For every immediate transition which represents the control structure relationship (eg. switch or while), we annotate to describe the weight assigned to the firing of enabled immediate transition t.

In this paper, the reliability measure of a web service we use is the probability of its ability to successfully carry out its own task when it is invoked. To associate the failure behaviour with the activities, we extend the CSPN model transformed from BPEL in section 3.2. For each transition representing the execution of an activity offered by a web service, two immediate transitions added to represent the events that results produced by the activity are correct and incorrect respectively, and have weights (the reliability of the web service) and . This process is depicted as Fig. 3, Place "Fail" represents the failure of the BPEL composite web service.

```
<flow>
  <link name="link1"/>
  <link name="link2"/>
  <link name="link3"/>
  <link name="link4"/>
  ...
  <activityX suppressJoinFailure="yes">
    <targets>
      <joinCondition>
        link1 AND link2
      </joinCondition>
        <target linkName="link1"/>
        <target linkName="link2"/>
    </targets>
    <sources>
      <source linkName=""/>
        <transitionCondition>
          P(...)
        </transitionCondition>
      <source linkName=""/>
        <transitionCondition>
          NOT P(...)
        </transitionCondition>
    </sources>
  <.activityX>
```

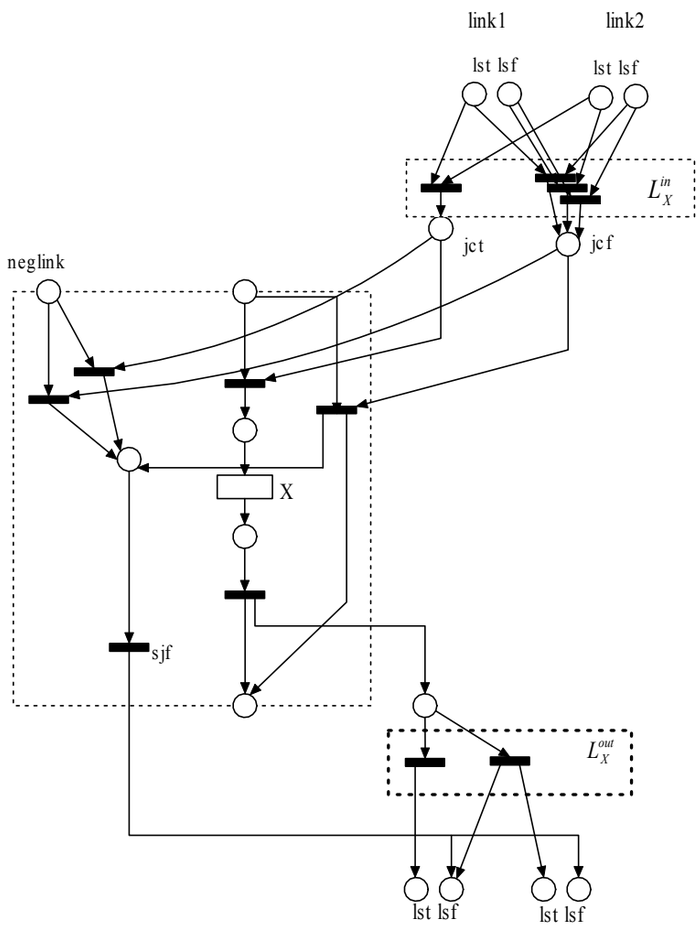


Fig. 2. Transformation of linked activity

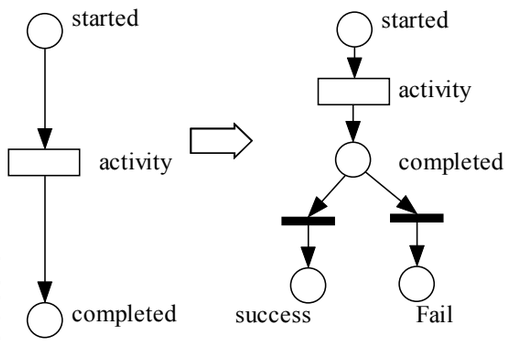


Fig. 3. Associate the failure behavior

The last step is to solve the stochastic Petri net model and compute the reliability prediction of web service composition. In this paper, we use the Stochastic Petri Net Package (SPNP) (C.Hirel et al., 2000) to computation of the reliability measures. SPNP is a versatile modelling tool for stochastic Petri net model; it allows the specification of SPN models, the computation of steady-state, transient, cumulative, time-averaged, and up-to-absorption measures and sensitivities of these measures. The most powerful feature of SPNP is the ability to assign reward rates at the net level and subsequently compute the desired measures of the system being modelled. Here we assign reward rate 1 to all markings in which there is no token in place "Fail"; all other markings are assigned a reward rate equal to zero. And the reliability of BPEL composite web service is the expected reward rate in steady state.

4. Examples

The following example shows how the structure of a BPEL process model is transformed into a stochastic Petri nets model. Fig.4 is the schematic illustration of the example taken from the section on structured activities of the BPEL 1.1 specification (BEA et al., 2003).

This example considers a simple loan approval web service that provides a port where customers can send their requests for loans. Customers of the services send their loan requests, including personal information and amount being requested. Using this information, the loan service runs a simple process that results in either a “loan approved” message or a “loan rejected” message. The approval decision can be reached in two different ways, depending on the amount requested and the risk associated with the requester. For low amounts (less than $10,000) and low-risk individuals, approval is automatic. For high amounts or medium and high-risk individuals, approval is to be studied in greater detail. The corresponding stochastic Petri nets model is depicted as Figure 5.

In this example, the following parameters must be assigned a value before the SRN model can be evaluated:

- the reliability of each partner
- the probability weights of the immediate transitions
- the execution time of each primitive activity

We assume the values given in Table 1. Using the SPNP 6.0, we compute the reliability prediction for the loan approval process as Rel=0.948=94.8%

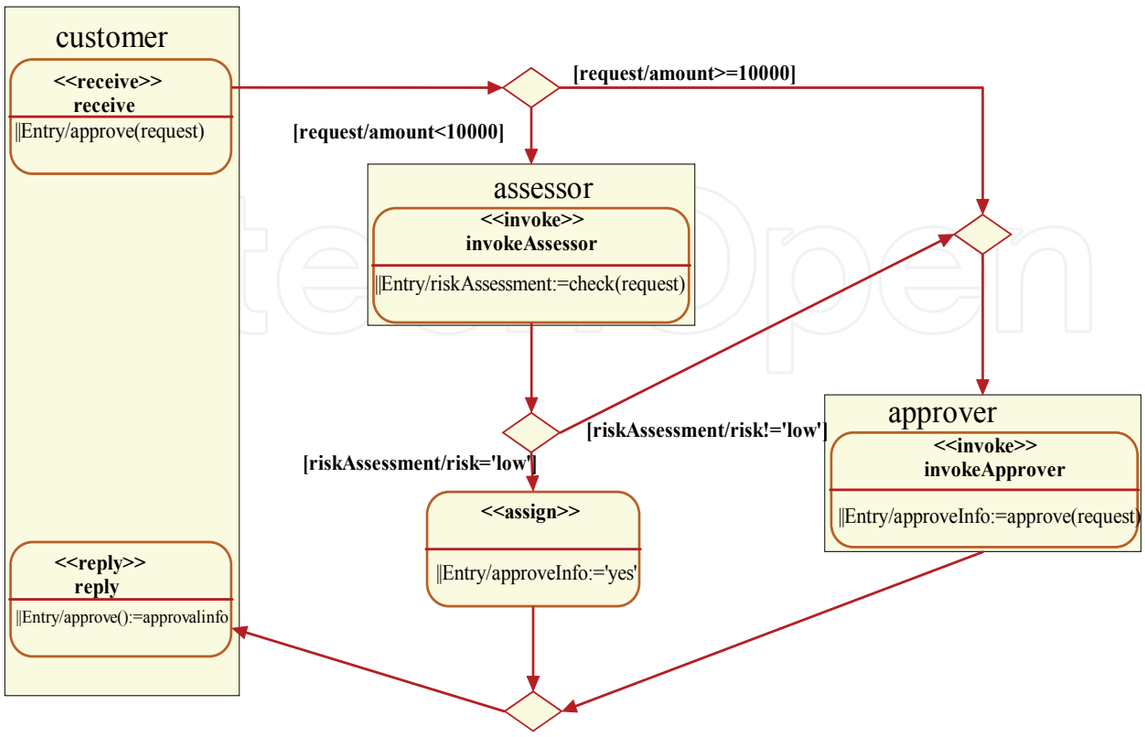


Fig. 4. Loan approval process

Reliability	Value
$R_{Customer}$	0.98
$R_{Assessor}$	0.99
$R_{Approver}$	0.99

Probability	Value
$pr\{amounts <= 10000\}$	0.4
$pr\{amounts > 10000\}$	0.6
$pr\{risk = low\}$	0.3
$pr\{risk = high\}$	0.7

Execution time	Value
$T_{receive}$	4
T_{reply}	4
T_{assign}	1
$T_{invoke\_Assessor}$	10
$T_{invoke\_Approver}$	15

Table 1. The parameters of loan approval process

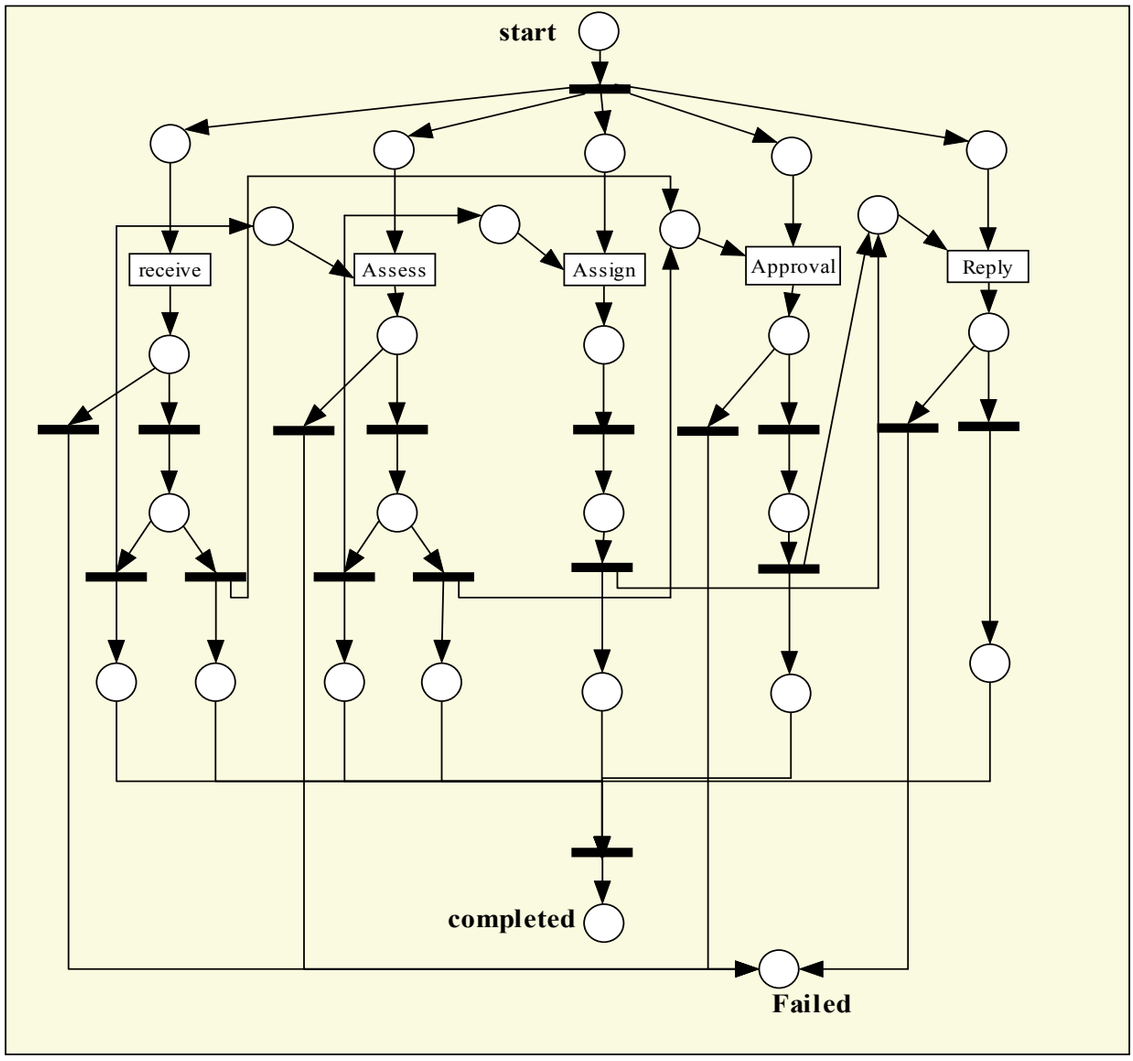


Fig. 5. The SPN model of loanapproval process

5. Sensitivity analysis

In this section, we illustrate some sensitivity analyses that can be performed for our reliability prediction technique: (1) as a function of the component service's reliability and (2) as a function of the usage profile. These analyses are exemplified using the loan approval process example.

5.1 System reliability as a function of component services' reliability

This analysis consists in varying the system reliability as a function of the component services' reliabilities with the purpose of identifying the component service that have the greatest impact on the reliability of the composite service. The method consists of varying the reliability of one component service at a time and fixing the others to 1. The probability distribution of the usage profile is same as section 4. Fig 6 shows the graphs of the reliability

of the composite service as a function of the component services' reliabilities. Note that the component service Approver has a large impact on the composite service's reliability, as the composite service invokes Approver more frequently than Assessor.

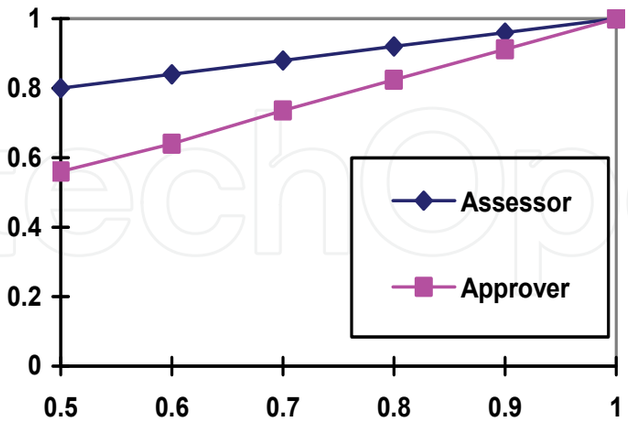


Fig. 6. Reliability as a function of component services' reliabilities

5.2 System reliability as a function of usage profile

This analysis consists in varying the system reliability as a function of the usage profile, as depicted in the graphs of Fig 7. The reliabilities of component services are same as section 4. If we vary the probabilities of low amounts and low risk from 0.3 to 0.9, the result is shown in Fig 7. Note that the usage profile does not have much impact on the system reliability, as the reliabilities between the component service Approver and Assessor are very close in this simple example.

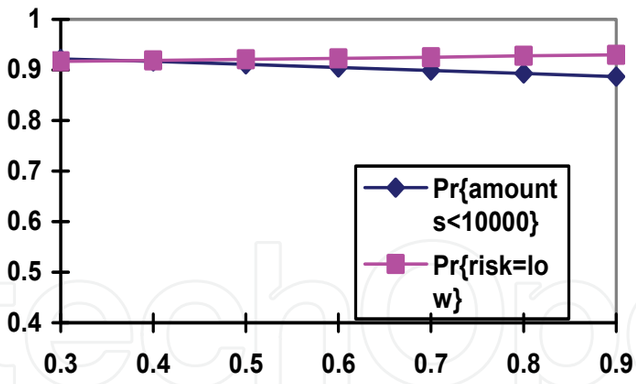


Fig. 7. Reliability as a function of service-usage profile

6. Related works

Approaches to the reliability analysis of service- and component-based system have been already presented. According to the classification proposed by Goseva Popstojanova (Goseva Popstojanova, 2001), they can be divided into two main categories: state-based approaches and path-based approaches. For the sake of brevity, we provide here a brief view of the approaches of greatest interest to the scope of this work.

State-based models (R.C.Cheung, 1980) use a control flow graph to represent the system architecture. In such models it is assumed that the transfer of control among the components

can be modelled as a Markov chain, with further behaviour of the system dependent only on the current state. The architecture of software has been modelled as a discrete time Markov Chain (DTMC), continuous time Markov Chain (CTMC), or a semi-Markov process (SMP). These can be further classified into absorbing and irreducible. The former represents applications that operate on demand which software runs that correspond to terminating execution can be clearly identified. The latter is well suited for continuously operating software applications, such that in real time control systems, where it is either difficult to determine what constitutes a run or there maybe very large number of such runs if it is assumed that each cycle consists a run.

Path-based models (S.M.Yacoub et al., 1999) compute the reliability of the system by enumerating possible execution paths of the program. The model used in their approach is the component dependency graph (CDG), this reliability analysis technique is specific for component based software whose analysis is strictly based on execution scenarios. A scenario is a set of component interactions triggered by specific input stimulus, and it is related to the concept of operations and run-types used in operational profiles (D.Musa, 1993).

Vincenzo Grassi present an approach to the reliability prediction of an assembly of services, that allows to take into account in an explicit and composition way the reliability characteristics of both the resources and interaction infrastructures used in the assembly (Vincenzo Grassi, 2005). What distinguishes their approach is the exploitation of a "unified " service model that helps in modelling and analyzing different architectural alternatives, where the characteristic of both "high level" services and "low level" services are explicitly taken into consideration. Moreover, this work also point out the importance of considering the impact on reliability of service sharing.

Apostolos focused on the development of a principled methodology for the dependability analysis of composite Web Services (Apostolos Zarras et al., 2004). The first step involves a UML representation for the architecture specification of composite web services. The proposed representation is built upon BPEL and introduces necessary extensions to support the dependability analysis. The automated mapping of this extended UML models to traditional dependability analysis models such as Block Diagrams, Fault Trees and Markov models is the core of the methodology.

7. Conclusion

In this paper, we introduce an approach to predict the reliability of Web services composition. We present the transformation algorithms from BPEL, which is the de facto industry standard of Web services composition specification, to CSPN models. Using the model, we can compute the reliability prediction of the web service composition. The major contribution of this paper is a reliability prediction technique that takes into account the structure of BPEL specification and the concurrent nature of service composition. For future work, we will transform all control-flow constructs of BPEL (including link, scope, faultHandler etc) into Petri nets. And we will use our CSPN model to give a more precise estimation of the reliability and performance of web service composition.

8. References

Axel Martens (2005), Analyzing Web Service based Business Processes. *In Proc. of FASE'05*, Edinburgh, Scotland.

Apostolos Zarras, Panos Vassiliadis, and Valerie Issarny(2004), Model-Driven Dependability Analysis of Web Services, *In Proceedings of the International Conference on Distributed Objects and Applications (DOA)*,LNCS3291.

BEA, IBM, Microsoft, SAP AG, and Siebel Systems (2003), Business process execution language for web services (version 1.1). ftp://www6.software.ibm.com/software/developer/library/ws-bpel.pdf.

C. Hirel, B. Tuffin, and K. S. Trivedi (2000), SPNP: Stochastic Petri Nets. Version 6.0, in Computer performance evaluation: Modelling tools and techniques, *11th International Conference; TOOLS 2000, Schaumburg, Il.*, USA, B. Haverkort, H. Bohnenkamp, C. Smith(eds.), LNCS 1786.

F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana(2002), Unraveling the web services web: An introduction to SOAP, WSDL, and UDDI, *IEEE Internet Computing*, 6(2). pp86–93.

Gianfranco Ciardo, Jogesh K. Muppala and Krishor S. Trivedi (1992), "Analyzing Concurrent and Fault-Tolerant Software using Stochastic Reward Nets", *Journal of Parallel and Distributed Computing, Vol. 15*, pp. 255-269.

H.M.W. Verbeek and W.M.P. van der Aalst(2005), Analyzing BPEL Processes using Petri Nets, *In Proceedings of the Second International Workshop on Applications of Petri Nets to Coordination, Workflow and Business Process Management,* Florida International University, Miami, Florida, USA, pp.59-78.

JD.Musa (1993), Opeartional profiles in software reliability engineering, In *IEEE Software 10(2).*

Jens Happe, Viktoria Firus (2005), Using Stochastic Petri Nets to Predict Quality of Services Attributes of Component-Based Software Architectures, *the Tenth International Workshop on Component-Oriented Programming*, Glasgow, Scotland.

K.Goseva-Popstojanova, A.P. Mathur, K.S.Trivedi(2001), Comparison of architecture-based software reliability models, *In Proc. Of the 12th Int. Symposium on Software Reliability Engineering (ISSRE 2001).*

Marsan A, Balbo G, Conte G, Donatelli S, Franceschinis G (1995), Modelling with Generalized Stochastic Petri Nets, Wiley, Chichester, England.

R.C.Cheung (1980), A User-Oriented Software Reliability Model, *In IEEE) Transactions on Software Engineering, volume 6(2)*, PP 118-125.

R.H. Reussner, H.W.Schmidit, I.H.Poernomo (2003), Reliability prediction for component-based software architectures., *Journal of Systems and Software,* no.66,pp241-252.

Sebastian Hinz, Karsten Schmidt, and Christian Stahl (2005), Transforming BPEL to Petri Nets, In *Proc. 3rd Int. Conf. on Business Process Management (BPM 2005)*, LNCS 3649, Nancy, France, pp. 220-235.

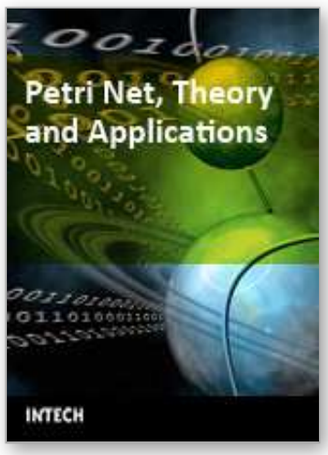
S.M.Yacoub,B.Cubic, and H.H.Ammar (1999), Scenario-Based Reliability Analysis of Component-Based Software, *In Proc. of the 10th ISSRE*, Boca Raton, FL, USA.

Simona Bernardi (2003), Building Stochastic Petri Net models for the verification of complex software systems, PHD Paper, Torino.

Vincenzo Grassi(2005), Architecture-Based reliability Prediction for Service-Oriented Computing, *Architecting Dependable Systems III*, LNCS 3549,pp.279-299.

W-L,Wang, Y.Wu, M-H Chen(1999), An Architecture-based software reliability model, *Proc. IEEE Pacific Rim Int. Symposium on Dependable Computing*, Hong Kong China.

Zhangxi Tan, Chuang Lin, Hao Yin, Ye Hong, and Guangxi Zhu(2004), Approximate Performance Analysis of web Services Flow Using Stochastic Petri Net, *In GCC 2004*, LNCS 3251,pp.193-200.



Petri Net, Theory and Applications

Edited by Vedran Kordic

ISBN 978-3-902613-12-7
Hard cover, 534 pages
Publisher I-Tech Education and Publishing
Published online 01, February, 2008
Published in print edition February, 2008

Although many other models of concurrent and distributed systems have been de- veloped since the introduction in 1964 Petri nets are still an essential model for concurrent systems with respect to both the theory and the applications. The main attraction of Petri nets is the way in which the basic aspects of concurrent systems are captured both conceptually and mathematically. The intuitively appealing graphical notation makes Petri nets the model of choice in many applications. The natural way in which Petri nets allow one to formally capture many of the basic notions and issues of concurrent systems has contributed greatly to the development of a rich theory of concurrent systems based on Petri nets. This book brings together reputable researchers from all over the world in order to provide a comprehensive coverage of advanced and modern topics not yet reflected by other books. The book consists of 23 chapters written by 53 authors from 12 different countries.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Duhang Zhong, Zhichang Qi and Xishan Xu (2008). Reliability Prediction and Sensitivity Analysis of Web Services Composition, Petri Net, Theory and Applications, Vedran Kordic (Ed.), ISBN: 978-3-902613-12-7, InTech, Available from: http://www.intechopen.com/books/petri_net_theory_and_applications/reliability_prediction_and_sensitivity_analysis_of_web_services_composition

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen